CLAIMS

What is claimed is:

1.   A method comprising:

      monitoring thread switches in a multiple-threaded application;

      executing a non-blocking thread synchronization sequence; and

      interrupting the non-blocking thread synchronization sequence upon the

occurrence of a thread switch.

2.   The method of claim 1 further comprising:

      repeating the non-blocking thread synchronization sequence.

3.   The method of claim 2 wherein the multiple-threaded applications are supported

by a computer programming language selected from the group consisting of JAVA, C#,

CLI, LISP, and Pascal.

4.   The method of claim 2 wherein the thread switches are monitored through use of

a thread switch flag.

5.   The method of claim 2 wherein the non-blocking thread synchronization sequence

is a frontier pointer-based allocation sequence.

6.   The method of claim 5 wherein executing the frontier pointer-based allocation

sequence comprises:

      loading a frontier pointer into a first register;

moving a current value of the frontier pointer to a second register;

adding the size of an object to be allocated to the first register such that a new

frontier pointer is determined;

storing a virtual method table to the second register if a thread switch has not

occurred; and

updating the frontier pointer with the new frontier pointer if a thread switch has

not occurred.

7. A machine-readable medium that provides executable instructions, which when

executed by a processor, cause the processor to perform a method, the method

comprising:

monitoring thread switches in a multiple-threaded application;

executing a non-blocking thread synchronization sequence; and

interrupting the non-blocking thread synchronization sequence upon the occurrence of a

thread switch.

8. The machine-readable medium of claim 7 further comprising:

repeating the non-blocking thread synchronization sequence.

9. The machine-readable medium of claim 8 wherein the multiple-threaded

applications are supported by a computer programming language selected from the group

consisting of JAVA, C#, CLI, LISP, and Pascal.

10.   The machine-readable medium of claim 8 wherein the thread switches are monitored through use of a thread switch flag.

11.   The machine-readable medium of claim 8 wherein the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.

12.   The machine-readable medium of claim 11 wherein executing the frontier pointer-based allocation sequence comprises:

loading a frontier pointer into a first register;

moving a current value of the frontier pointer to a second register;

adding the size of an object to be allocated to the first register such that a new frontier pointer is determined;

storing a virtual method table to the second register if a thread switch has not occurred; and

updating the frontier pointer with the new frontier pointer if a thread switch has not occurred.

13.   A computing system comprising:

at least one central processing unit, the central processing unit executing multi-threaded applications;

a thread switch indicator to indicate the occurrence of a thread switch; and

an instruction set to implement non-blocking thread synchronization sequences such that partially completed non-blocking thread synchronization sequences used to

share resources local to the at least one central processing unit can be abandoned and repeated upon the occurrence of a thread switch.

14. The computing system of claim 13 wherein the instruction set includes:

a set instruction to set the thread switch indicator upon the occurrence of a thread switch;

a first conditional move instruction to move data if the thread switch indicator is set;

a second conditional move instruction to move data if the thread switch indicator is not set;

a first jump instruction to bypass instructions if the thread switch indicator is set;

a second jump instruction to bypass instructions if the thread switch indicator is not set; and

a clear instruction to clear the thread switch indicator.

15. The computing system of claim 14 wherein the thread switch indicator is a thread switch flag.

16. The computing system of claim 13 wherein each of the at least one central processing units has a single allocation area and the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.

17. The computing system of claim 13, wherein the computing system uses a computer programming language selected from the group consisting of JAVA, C#, CLI, LISP, and Pascal.

18. A computer system instruction set comprising:

a thread switch indicator to indicate the occurrence of a thread switch;

a set instruction to set the thread switch indicator upon the occurrence of a thread switch;

a first conditional move instruction to move data if the thread switch indicator is set;

a second conditional move instruction to move data if the thread switch indicator is not set;

a first jump instruction to bypass instructions if the thread switch indicator is set;

a second jump instruction to bypass instructions if the thread switch indicator is not set; and

a clear instruction to clear the thread switch indicator.

19. The computer system instruction set of claim 18 implemented as hardware.

20. The computer system instruction set of claim 18 wherein the thread switch indicator is a thread switch flag.

21. The computer system instruction set of claim 18 used to implement a non-blocking thread synchronization sequence for the execution of multi-threaded applications.

22. The computer system instruction set of claim 21 wherein the non-blocking thread synchronization sequence is a frontier pointer-based allocation sequence.